Security of Distributed Information Systems Based on Component-Based Software Engineering

Yassine CHAOUCHE, Youssef FAKHRI

Abstract— Distributed systems have the particularity of linking several machines that coexist in a heterogeneous and unreliable environment. Moreover, their proper functioning depends on the messages they share on the network. It was therefore essential to question the security measures adapted and deployed in this context. We will present in this article the most known security threats that these systems suffer then we will track the different mechanisms of securing distributed systems, then we will present the CBSE model as an appropriate solution to the requirements of modularity, dynamicity and security of this type of systems.

Index Terms— Security, Distributed system, Security Threats, Encryption, Cryptography, CBSE, Fractal .

1 Introduction

Distributed systems (or distributed systems), as opposed to centralized systems, are composed of several independent machines, having distinct physical memories, connected to each other in a network and communicating via this network. From the point of view of the user, no difference is perceptible between a distributed system and a centralized system. That said, these systems make it possible to guarantee properties that are not available in a centralized system, for example redundancy, which makes it possible to mitigate material faults or to make the same service available to several actors without loss of time; performance, guaranteed by the pooling of several computing units allowing parallel processing in a shorter time; and data protection, which is not available everywhere at the same time, but only some views are exported.

A distributed system is usually separable into several fully autonomous modules, each responsible for its own operation. This autonomy makes it possible on the one hand to use heterogeneous technologies, platforms or languages in each of these modules, and on the other hand to run them simultaneously and thus guarantee a concurrent programming.

However, distributed systems are subject to several risks, due to the points of failure they possess in addition to centralized systems, such as the unsecured network, traffic, nodes themselves, and so on.

In the following sections, we present the various security threats that distributed systems may encounter as well as the different mechanisms used to avoid them, then we will introduce the Component-based software engineering (CBSE) model as a solution for separately implementing nodes and communication channels, while meeting the requirements of modularity, dynamicity and security of this type of system.

2 Security Threats for Distributed Systems

The most well-known security threats for distributed systems are caused by network attacks, the most common of which are:

Distributed Denial of Service (DDoS) is an attack that renders the service unavailable to users by overwhelming it with unnecessary traffic. This attack is caused by several machines at once (unlike the DoS, which is perpetrated by a single attacker), and is difficult to counter or avoid.

MITM (Man In The Middle) This attack of the Middle Man is a form of espionage in which the attacker makes independent connections with the victims and relays the messages between them, making them believe that they are talk to each other The attacker can thus intercept all the messages circulating between the victims and inject new ones, by passing themselves respectively by one of the victims to the other.

IP Spoofing This is an attack where the attacker personifies another machine by sending messages with his IP address.

Packet Sniffing This is an attack where the attacker intercepts and records traffic flowing through the network.

Replay Attack This is an attack by the Middle Man where the attacker repeats or delays a valid data transmission. It can be useful for the attacker in the case where, for example, he wishes to impersonate a user by saving his encrypted password used in a first exchange as proof of identity, and returning it to another exchange.

These attacks are quite common, and their advent can be somewhat dangerous for the system, especially if the manipulated data is critical, such as bank data or personal information. To avoid or counteract them, we must ensure certain security properties. We quote in this part the most important properties.

3 Usual security properties

The most common security properties for distributed systems are: Authentication This property represents the procedure that allows the distributed system to verify the identity of an entity, whether a user or a machine that is part of or not part of the system. to allow access to resources. It thus makes it possible to validate the authenticity of the entity in question.

Confidentiality This property makes it possible to protect information whose access is limited to only the entities that are known to know it.

Integrity This property implies that the alteration of

information can only be done in a voluntary and legitimate way.
Availability This property guarantees the ability of the system to perform a function under defined schedule, time and performance conditions.
Non repudiation This property ensures that the information can not be plausibly disavowed.

4 SECURITY MECHANISMS FOR DISTRIBUTED SYSTEMS

To be able to protect the distributed systems from the attacks mentioned above, security mechanisms are defined. We quote them in this part.

4.1 Access control

Access control is a security mechanism designed to protect physical resources by verifying whether an entity requesting access to this resource has the necessary rights to do so.

To provide access control, the system must provide both an authentication mechanism, which allows an entity to be recognized by the system (for example, a password or a card), and a mechanism for accessing the system. authorization, which allows associating with an entity a set of rights on a resource.

According to Nikander[7] access control includes the concept of an access control matrix, where the columns carry the names of the subjects (active entities), the lines of the objects, and each cell includes the actions that the subject is authorized to perform on the object. In practice, the access control matrix is an abstract elemaratelent. The information that is included is usually shown sepy line by line, in the form of ACLs (Access Control Lists), or column by column in the form of capabilities.

4.2 Cryptographic primitives

Since distributed systems have the particularity of evolving in an often unreliable environment, all the data exchanged between the nodes must be secured at the application level. This security during transport implies that (1) the secret data must not be visible to an attacker, (2) the integrity of the data must be preserved, in the sense that the data must not be modified during transport by third parties and (3) the receiver of the data must be able to verify that this data comes from the entity that claims to have sent it. To be able to guarantee these properties, cryptographic mechanisms are generally used.

4.2.1 Hash function (Cryptographic hash)

A hash function is a function that ensures the integrity of a message. It takes a message as input and generates a block of bits, of length reaching several hundred bits, which represents the digital fingerprint of the message (message digest). If the message is modified, even slightly, by a third person for example, a significant change is made in the fingerprint (ideally, 50% of the fingerprint changes for a bit changed in the initial message).

Several hashing algorithms are defined. The following two algorithms are the most used:

- MD6, for Message Digest 6, is a cryptographic hash function that allows to obtain the digital fingerprint of a file (we often speak of message). MD6 was developed by a group1 led by Ronald L. Rivest, American cryptologist who invented MD5 and participated in the development of RSA, with Shamir and Adleman.

MD6 was proposed to participate in the NIST hash function competition in 2008 but was not selected in the second selection stage.

SHA-2 (Secure Hash Algorithm) is a family of hash functions that have been designed by the US National Security Agency (NSA), modeled on the SHA-1 and SHA-0 functions, which are themselves strongly inspired. of Ron Rivest's MD4 function (which paralleled MD5).

4.2.2 Encryption

4.2.2.1 Symmetric encryption

The symmetric encryption is as follows: Alice and Bob each have a shared key that they are the only ones to know. They agree to use a common cryptographic algorithm, called a cipher. When Alice wants to send a message to Bob, she encrypts the original message (plain text) to create a cryptogram. She then sends the cryptogram to Bob, who receives it and decrypts it with his secret key to recreate the original clear message. If Chuck is spying on their communication, he can only see the cryptogram. Thus, the confidentiality of the message is preserved.

It is possible to encrypt bit by bit or block by block. The blocks are typically 64 bits in size. If the message is not a multiple of 64, then the last block (the shortest) must be filled with random values until it reaches 64 bits (this concept is called padding). Bit-by-bit encryption is no longer used for hardware implementations.

The strength of private key encryption is determined by the cryptography algorithm and the length of the key.

There are several algorithms for private key encryption, including:

- DES (Data Encryption Standard): invented by IBM in 1970 and adopted by the American government as standard. It is a 56-bit block algorithm.
- TripleDES: Used to handle 56-bit key loopholes by increasing the DES technology by passing the plaintext through the DES algorithm 3 times, with two different keys, giving the key a real strength of 112 bits . Also known as DESede (for encrypt, decrypt and encrypt, the three phases through which it passes).
- AES (Advanced Encryption Standard): replaces DES as American standard. It was invented by Joan Daemen and Vincent Rijmen and is also known as the Rinjdael algorithm. It is a 128-bit block algorithm with keys of length 128, 192 or 256 bits.
- Blowfish: Developed by Bruce Shneider. It is a variable key length algorithm ranging from 32 to 448 bits (multiples of 8), and used mainly for an implementation on software for microprocessors.
- PBE (Password Based Encryption): algorithm that uses the password as an encryption key. It can be used in combination with a variety of digital fingerprints and private key

algorithms.

4.2.2.2 Asymmetric encryption

Symmetric encryption suffers from a major drawback: how to share the key between Alice and Bob? If Alice generates it, she has to send it to Bob; however, it is sensitive information that needs to be encrypted. However, no key has been exchanged to achieve this encryption.

Asymmetric encryption, invented in the 1970s, solved the problem of encrypting messages between two parties without prior agreement on the keys. In this type of encryption, Alice and Bob each have two different pairs of keys: a key is secret and should not be shared with anyone, and the other is public, and therefore visible to everyone.

If Alice wants to send a secret message to Bob, she encrypts the message using Bob's public key and sends it to him. Bob then uses his private key to decipher the message. Chuck can see the two public keys as well as the encrypted message, but can not decipher the message because he does not have access to the secret keys.

The keys (public and secret) are generated in pairs and are larger than the equivalent private encryption keys. It is not possible to deduce a key from the other. The following two algorithms are used for public key encryption:

- RSA: (Rivest Shamir Adleman): described in 1977 by Ronald Rivest, Adi Shamir and Leonard Adleman, and patented by MIT (Massachusets Institute of Technology) in 1983. This is the most popular public key encryption algorithm . It is widely used in e-commerce and Internet data exchange in general.
- Diffie-Hellman: technically known as a key-agreement algorithm: it can not be used for encryption, but to allow both parties to deduce a secret key by sharing information on a public channel. This key can then be used for symmetric encryption.

4.2.2.3 Session Key Encryption

Public key encryption is slow (100 to 1000 times slower than private key encryption), so a hybrid technique is typically used in practice. One of the parties generates a secret key, called a session key, which it encrypts with the public key of the other party to send to it. Then symmetric encryption is used to encrypt the message using the exchanged session key.

4.2.3 Signature

The problem with encryption is above all to prove that the message comes from the sender who claims to have sent it. Chuck could send a request to Bob pretending to be Alice

This problem can be solved using the digital signature. It is a model used to prove that a message comes from a given part. One way to implement a digital signature is to use the reverse process with asymmetric encryption. Instead of encrypting the message with the public key and decrypting it with the private key, the private key is used by the sender to sign the message and the recipient uses the public key of the sender to decrypt it. As only the sender knows the private key, the recipient can be sure that the message really comes from him.

In reality, only the digital fingerprint (instead of the whole message) is signed by the private key. Thus, if Alice wants to send Bob a signed message, she generates the message's print and signs it with her private key. She sends the message (in plain text) and the signature to Bob. Bob decrypts the signed fingerprint with Alice's public key, calculates the fingerprint from the plaintext message, and verifies that both fingerprints are identical. If so, Bob is assured that it was Alice who sent the message.

Note here that the digital signature does not encrypt the message, so encryption techniques should be used in conjunction with signatures if confidentiality is also required.

4.2.4 certificates

The use of the digital signature can prove that a message was sent by a given party, but how to make sure that the public key used as that of Alice is not really that of Amanda? This problem can be solved by using a digital certificate, which makes it possible to package an identity with the public key and which is signed by a third party called Certificate Authority (CA).

A certification authority is an organization that verifies the identity (in the sense real physical identity) of a party and signs the public key and identity of that party with its private key. The recipient of a message can obtain the digital certificate of the sender and verify (or decrypt) with the public key of the CA, previously known by all parties. This proves that the certificate is valid and allows the recipient to retrieve the public key from the sender to verify its signature and send it an encrypted message.

4.3 Delegation

Delegation is a security mechanism that allows a subject to delegate their permissions and rights to another subject. When an entity is approved for delegation, it can represent the other entity and use services on its behalf. Delegation is useful if you want to optimize the number of stored identities, or avoid systematic recourse to the CA.

There are two types of delegation:

- Authentication delegation: It is defined whether an authentication mechanism provides an identity different from the valid identity of the user, provided that the owner of the effective identity has already authorized the other user to use your own identity.
- Access Control Delegation: This is done when a user delegates some of his permissions to another to access a resource.

Delegation has been implemented in several ways in the literature. For example, we quote Welch[8] who defines delegation as the authorization given to a user to dynamically assign a new X509 identity to an entity and thereby delegate some of its rights to it. Users create a proxy-type certificate by issuing an X509 certificate signed by their own claims instead of dealing with a CA. Proxy certificates can build trust domains dynamically, assuming that two entities that use proxy certificates issued by the same authority trust each other.

Nikander[7] attests that the delegation means "to give someone the power to act as a representative". This can change the access control matrix. The delegation is carried out thanks to a certificate SPKI (Simple Public Key Infrastructure),

represented by 5 fields: C = (I, S, D, A, V)

- **Issuer** (I for Issuer): the authority that created and signed the certificate. Represented by its public key or hash.
- **Subject** (S for Subject): Party for whom the certificate is issued.
- **Authority** (A for Authority): Semantic content specific to the application representing the authority.
- **Delegate?** (D for Delegated?): Can this authority in the certificate be delegated to someone else?
- **Validity** (V for Validity): When is the certificate valid? (period, URL of an online verification service ...)

The issuer delegates right A to the subject S. If S is a public key and D is true, then S can delegate that right to someone else. The validity of the delegation is limited by V. The node operating system is the only source of primary authority in the system. The person or system that installs the operating system for the first time has the ability to create initial delegations of that authority (equivalent to establishing an administrator account with a password).

All the security mechanisms we have presented are designed to ensure good communication between the different nodes of a distributed system by securing the transport of messages. However, the risks of information disclosure are not limited to communication channels, but can sometimes occur at a node itself. The idea would be to guarantee security both between the nodes and within the same node. It is for this reason that it is important to use an explicit representation for distributed systems that can separately implement nodes and communication channels, while meeting the requirements of modularity, dynamicity and security of this type of systems. CBSE (Component-based software engineering) model seems the most appropriate for this task

5 CBSE (COMPONENT-BASED SOFTWARE ENGINEERINGSECTIONS)

CBSE (Component-based software engineering) is a branch of software engineering that enables the separation of concerns according to the functionalities available in a given software system, thanks to its decomposition into components.

A component is a composition unit that can be deployed independently andassembled with other components. Thanks to their modularity, the components simplify the development and management of distributed systems.

Several studies have shown the role of the component in automating the management of distributed systems [Abdellatif07, Beisiegel05, Broy98]. Broy et al. define the components as entities that must (1) encapsulate data, (2) be implantable into most programming languages, (3) be hierarchically interlaced, (4) have clearly defined interfaces and (5) be able to be embedded in frameworks.

The architecture is described in an Architecture Description Language (ADL). The system is then automatically deployed on the hosts distributed. Each component can be configured separately through configuration interfaces (or attributes) that provide values for component attributes.

We distinguish for each component the server ports, which receive information.

other components, client ports, which issue information as messages. In addition, the components are weakly coupled, which means that the connections between the different components-called links-can be established in different ways, regardless of the component code. A link is established between a client port and a port server.

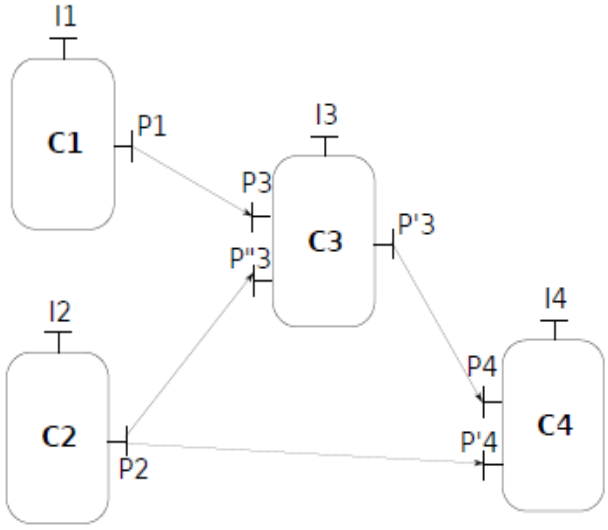

Figure1: Example of a component-based system

Figure I shows an example of a component-based system. C1, C2, C3 and C4 are components. Each component Ci admits two types of interfaces: the interfaces of Ii control to configure the component and the communication ports Pi allowing sending messages from one component to another. Communication ports can be connected by explicit links. In this example, C1 is connected to C3, which means that C1 can send data to C3 via port P1, and C3 receives it via its port P3. There are two types of ports: client ports that send requests and ports servers that receive them. For example, in component C3, P3 is a server port and PO3 is a client port. By convention, server ports are shown to the left of the component, and the client ports on his right. A client port can be attached to multiple server ports, such as for example the P2 port to the PO ports 4 and P "3. This implies that the same query is sent to C3 and C4.

5-1 Examples of component-based models

Component-based engineering has been used in several works to define a model for the construction of the systems. The models we are interested in are those that allow the clear separation of architecture and implementation in different structures. Several models correspond to this description, we chose the Fractal model.

5-1-1 FRACTAL

Fractal was produced by INRIA and the France Telecom R & D unit in June 2002. Fractal is a modular and extensible software component model for building systems distributed highly adaptable and reconfigurable. It is used to implement, deploy and reconfigure systems and applications.

A Fractal component consists of two parts: a controller, also called a membrane and a content. Figure II shows a simple

example of a system modeled with Fractal.

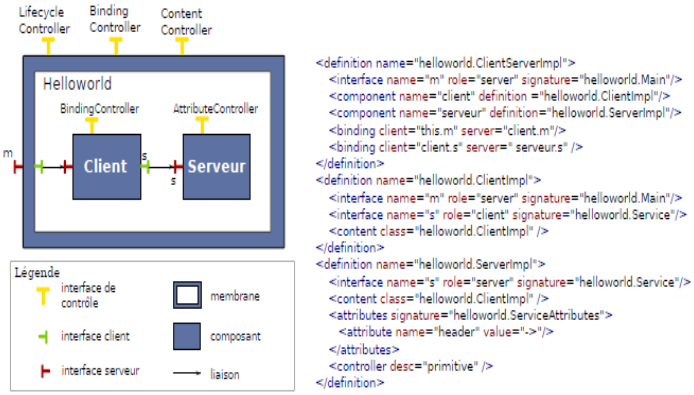

Figure 2: Example of an architecture and ADL Fractal

contents

The content of a component is composed of a finite number of other components, called subcomponents, which are under the control of the container component controller. The model Fractal is thus recursive. A component that exposes its content is called composite. A component that does not expose its content but that has at least one control interface is called primitive component. A component without a control interface is called a component basic.

Controller

The controller of a component can have external interfaces, accessible from the outside component or internal, accessible only from subcomponents. An interface functional is an interface that corresponds to a provided functionality (server interface) or required (client interface) of a component, whereas a control interface is a server side which corresponds to a non functional aspect.

The control interfaces are listed in several categories:

- AttributeController: The interface responsible for managing the attributes of a component. A attribute is a configurable property of a component, usually of primitive type, and used to configure the state of a component.
- BindingController: Interface responsible for managing the links of the interfaces other components.
- ContentController: Interface responsible for managing the subcomponents of a component site.
- LifeCycleController: The interface responsible for managing the execution of a component, starting, stopping, adding and removing subcomponents, links or attributes, dynamic way.

link

A link is a communication path between component interfaces. The model Fractal distinguishes between primitive and composite links. A primitive link is a link between a client interface and a server interface of the same address space.

A composite link is a communication path between a random number of interfaces of components and types of languages. These links are represented by a set of links primitives and communication components (also called connectors).

6 Conclusion

The purpose of this article is to show the importance of the component-oriented paradigm as a facilitator for the application and verification of an information flow control property, called the non-interference we have presented the CBSE model as being an appropriate solution to the requirements of modularity, dynamicity and security of distributed information systems by giving the example of the fractal model invented by INRIA, the model presented can be applied even on large and critical distributed systems, where any leak of information can be fatal, such as electronic voting systems, which have the particularity to be highly dynamic and any leakage can affect the anonymity of participants.

References

[1] Ptitsyn, P. S., & Radko, D. V. (2015). An analysis of technologies for building information security infrastructure of global

[2] distributed computing systems. Journal of Theoretical and Applied Information Technology, 82 (1), 45-53

[3] Grusho, Alexander, Grusho, Nick, Levykin, Michael, & Timonina, Elena. (2017). Analysis of information security of distributed information systems. Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2017 9th International Congress on, 2017, 96-100

[4] https://www.tutorialspoint.com/cryptography/public_key_encryption.htm

[5] http://gleamly.com/article/introduction-attribute-based-encryption-abe

[6] https://www.geeksforgeeks.org/rsa-algorithm-cryptography

[7] P. Nikander. An architecture for authorization and delegation in distributed-object-oriented agent systems. Citeseer, 1999. 13, 17, 41, 43

[8] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. In Pro-eedings. 12th IEEE International Symposium on High Performance Distributed Computing, 2003., pages 48-57. IEEE Comput. Soc., 2003. 17, 41, 42, 46

[9] en.wikipedia.org/wiki/List_of_hash_functions